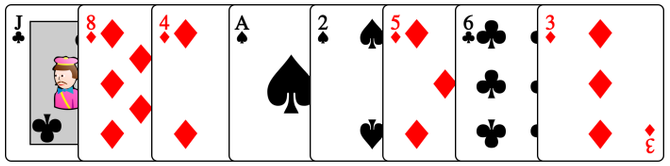
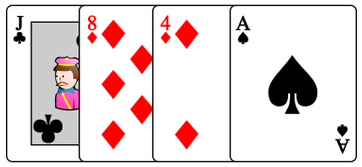
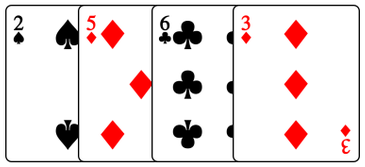
# Introduction à l’informatique CM5

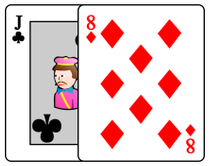
Antonio E. Porreca
https://aeporreca.org/teaching

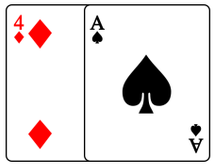
# Tri fusion

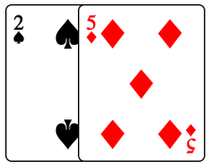


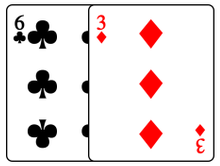


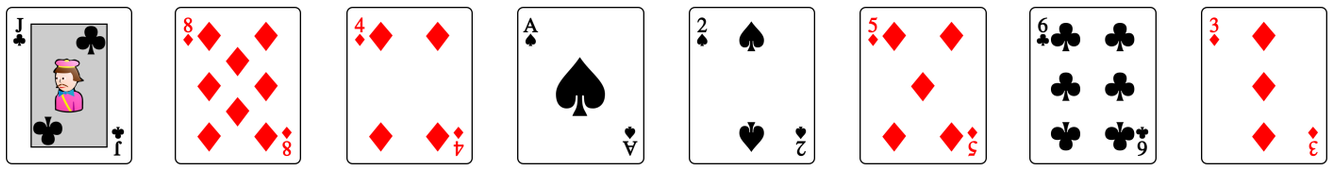


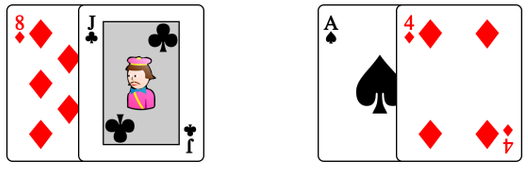


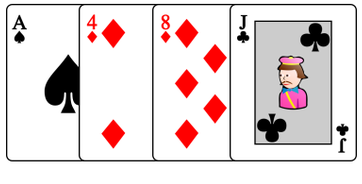


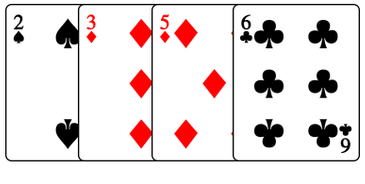


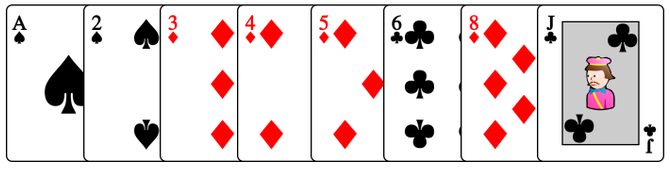




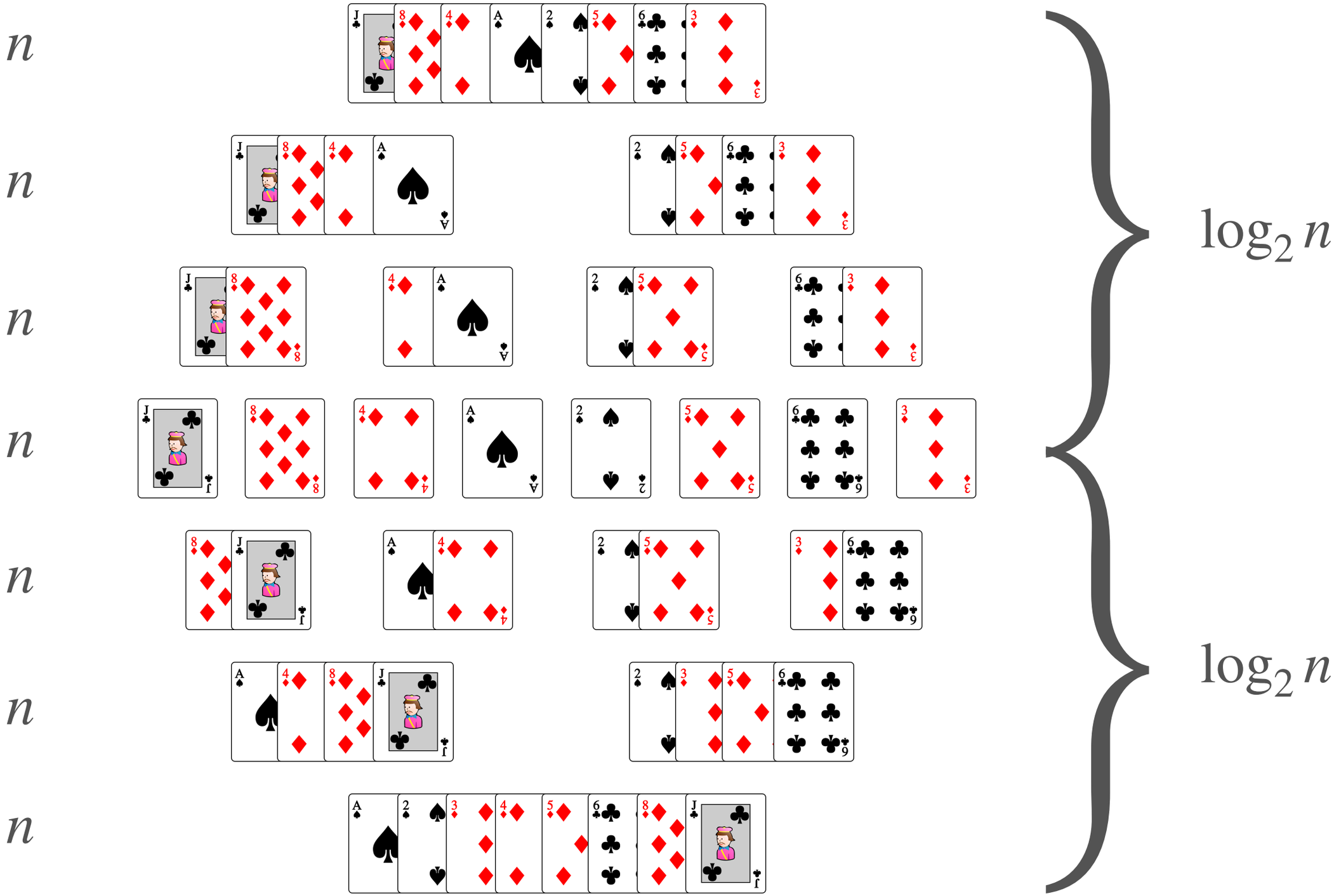








# Efficacité du tri fusion



# Fusion

```
fonction fusionner(A, B)
    n ≔ longueur(A)
    m ≔ longueur(B)
    i ≔ j ≔ k ≔ 0
    tant que i < n et j < m faire
        si A[i] < B[j] alors
            C[k] ≔ A[i]
            i ≔ i + 1
        sinon
            C[k] ≔ B[j]
            j ≔ j + 1
        k ≔ k + 1
    tant que i < n faire
        C[k] ≔ A[i]
        i ≔ i + 1
        k ≔ k + 1
    tant que j < m faire
        C[k] ≔ B[j]
        j ≔ j + 1
        k ≔ k + 1
    retourner C
```

encore d'éléments dans A et B

encore d'éléments dans A, mais B épuise

encore d'éléments dans B, mais A épuisé

A et B épuisés

# Tri fusion

```
procédure tri-fusion(A)
    n ≔ longueur(A)
    si n > 1 alors
        m ≔ ⌊n ÷ 2⌋
        B ≔ A[0, …, m]
        C ≔ A[m+1, …, n–1]
        B’ ≔ tri-fusion(B)
        C’ ≔ tri-fusion(C)
        retourner fusionner(B’, C’)
    sinon
        retourner A
```

# Algorithmes sur les entiers

# Incrémentation

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

# Incrémentation

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

+1

# Incrémentation

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

+1

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | 0 |

# Incrémentation

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

+1

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | 0 | 0 |

# Incrémentation

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

+1

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | 0 | 0 | 0 |

# Incrémentation

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

+1

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | 0 | 0 | 0 | 0 |

# Incrémentation

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | 1 | 0 | 0 | 0 | 0 |

# Incrémentation

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

# Dépassement d'entier (overflow)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

# Dépassement d’entier (overflow)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

+1

# Dépassement d'entier (overflow)

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

+1

| | | | | | | | 0 |
|---|---|---|---|---|---|---|---|

# Dépassement d'entier (overflow)

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

+1

| | | | | | | 0 | 0 |
|---|---|---|---|---|---|---|---|

# Dépassement d'entier (overflow)

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

+1

| | | | | | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

# Dépassement d'entier (overflow)

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

+1

| | | | | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

# Dépassement d'entier (overflow)

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

+1

| | | | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

# Dépassement d'entier (overflow)

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

+1

| | | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

# Dépassement d'entier (overflow)

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

+1

| | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

# Dépassement d'entier (overflow)

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

+1

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

# Dépassement d'entier (overflow)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Incrémentation

```
fonction incrémenter(N)
    i ≔ longueur(N) – 1
    tant que i > 0 et N[i] = 1 faire
        M[i] ≔ 0
        i ≔ i – 1
    si i ≥ 0 alors
        M[i] ≔ 1
    retourner M
```

# Addition

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | + |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | = |
| | | | | | | | | |

# Addition

+1

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | + |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | = |
| | | | | | | | 0 | |

# Addition

+1

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | + |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | = |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | 1 | 0 |

# Addition

+1

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | + |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | = |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | 1 | 1 | 0 |

# Addition

+1

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | + |

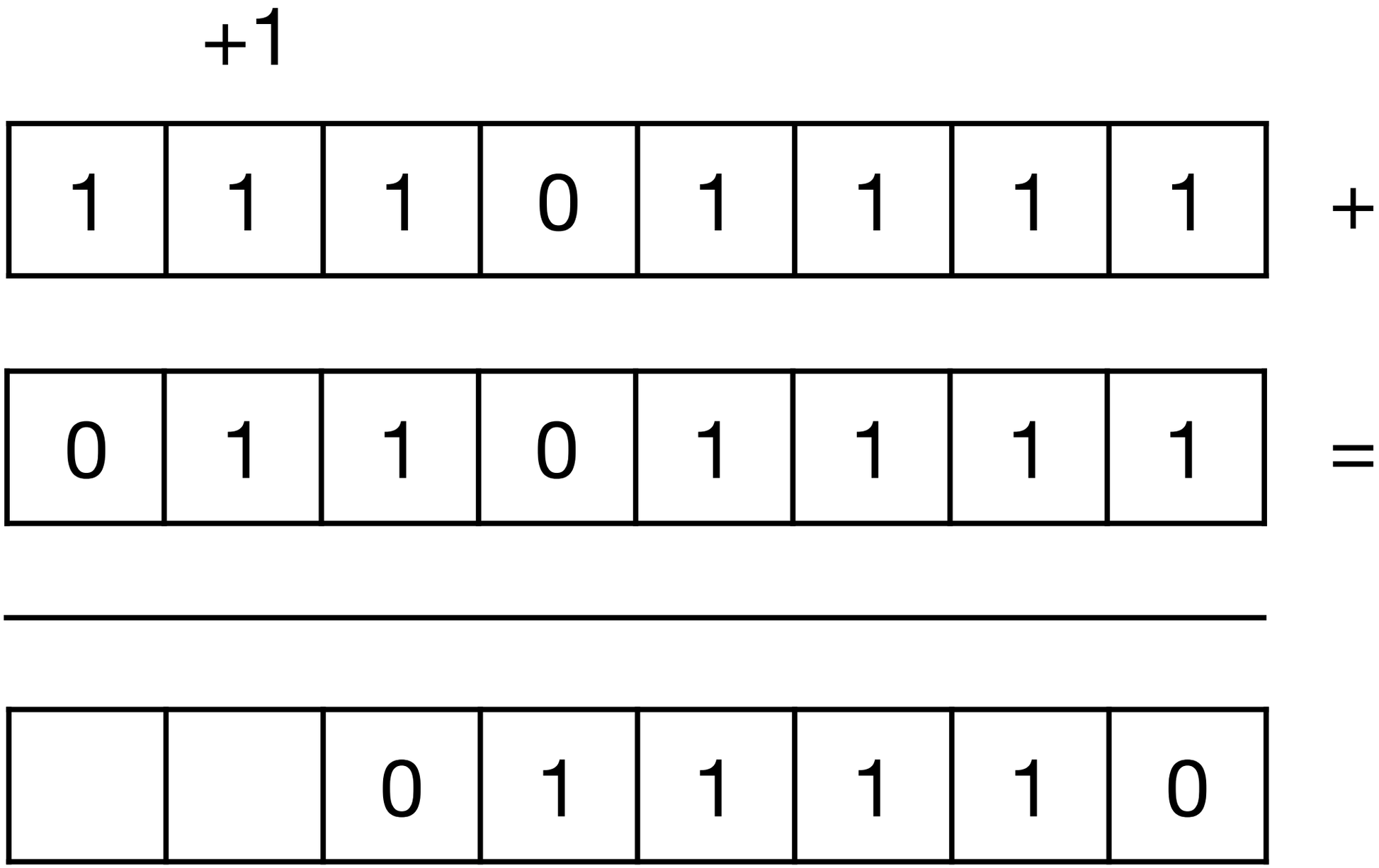
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | = |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | 1 | 1 | 1 | 0 |

# Addition

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | + |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | = |
| | | | 1 | 1 | 1 | 1 | 0 | |

# Addition



# Addition

+1

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | + |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | = |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

# Addition

+1

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | + |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | = |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

# Addition

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | + |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | = |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | |

# Division euclidienne d'*a* par *b*

# Division euclidienne d'*a* par *b*

$$a = q \times b + r \qquad \textbf{avec } 0 \leq r < b$$

# Division euclidienne d'*a* par *b*

$$a = q \times b + r$$

quotient reste

**avec** $0 \leq r < b$

# Division euclidienne

```
fonction division-euclidienne(a, b)
    q ≔ 0
    r ≔ a
    tant que r ≥ b faire
        q ≔ q + 1
        r ≔ r – b
    retourner (q, r)
```

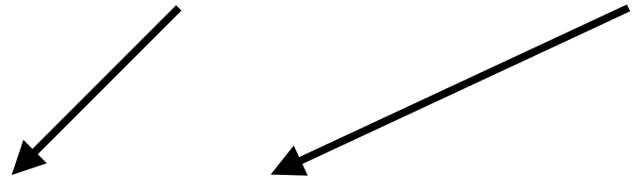
# Plus grand diviseur commun

# Plus grand diviseur commun

$$21 = 14 \times 1 + 7$$

# Plus grand diviseur commun

$$21 = 14 \times 1 + 7$$



$$14 = 7 \times 2 + 0$$

# Plus grand diviseur commun

$$21 = 14 \times 1 + 7$$

$$14 = 7 \times 2 + 0$$

$$\text{pgdc}(21, 14) = 7$$

# Plus grand diviseur commun

$$799 = 345 \times 2 + 109$$

# Plus grand diviseur commun

$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

# Plus grand diviseur commun

$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

$$109 = 18 \times 6 + 1$$

# Plus grand diviseur commun

$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

$$109 = 18 \times 6 + 1$$

$$18 = 1 \times 18 + 0$$

# Plus grand diviseur commun

$$799 = 345 \times 2 + 109$$

$$345 = 109 \times 3 + 18$$

$$109 = 18 \times 6 + 1$$

$$18 = 1 \times 18 + 0$$

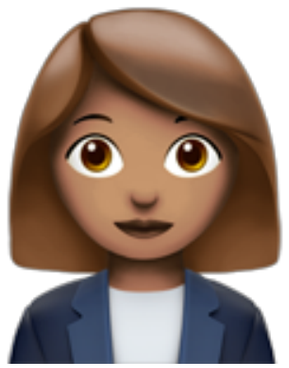
$$\text{pgdc}(799, 345) = 1$$

# Algorithme d'Euclide

**fonction** pgdc(a, b)
    (*a ≥ b entiers ≠ 0*)
    r ≔ a **mod** b
    **tant que** r > 0 **faire**
        a ≔ b
        b ≔ r
        r ≔ a **mod** b
    **retourner** b

# Ça sert à quoi ?

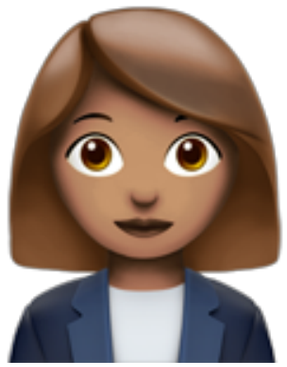
# Communications sécurisées



Alice

Bob

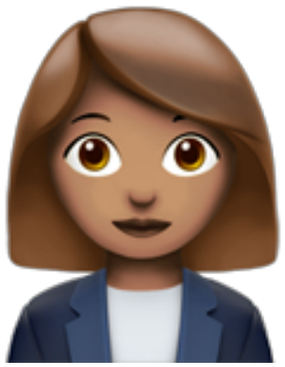
# Communications sécurisées



Alice

Bob

# Communications sécurisées

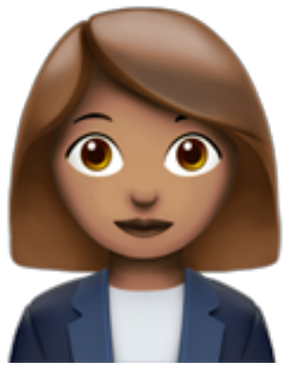


Alice

Bob

Eve

# Communications sécurisées

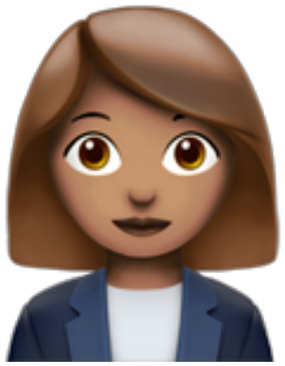


Alice

Bob

Eve

# Communications sécurisées

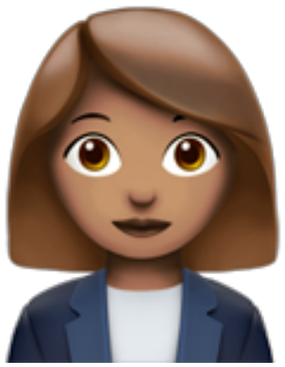


Alice

Bob

Eve

# Communications sécurisées

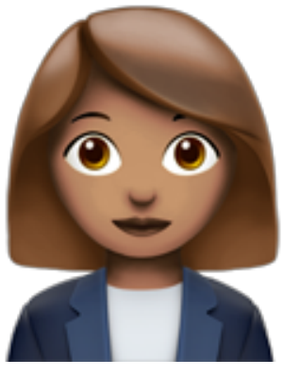


Alice

Bob

Eve

# Communications sécurisées

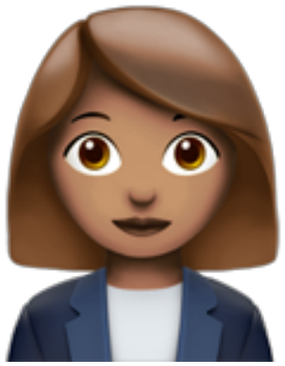


Alice

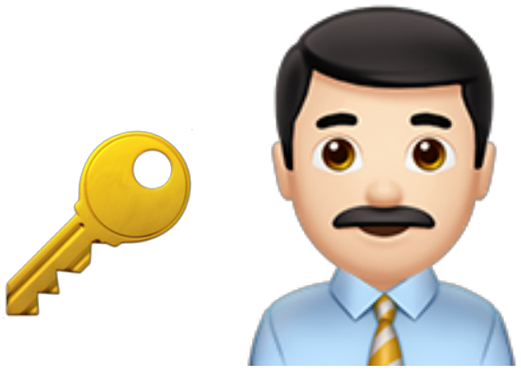
Bob

Eve

# Communications sécurisées



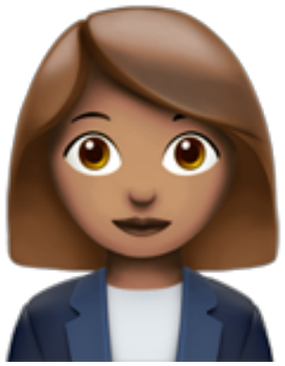
Alice



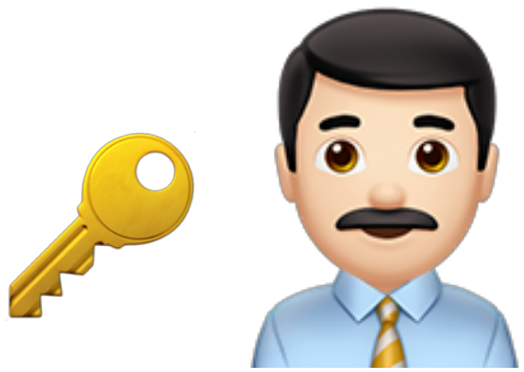
Bob

Eve

# Communications sécurisées



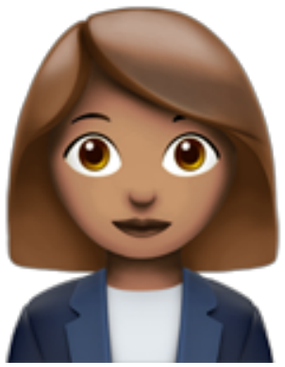
Alice



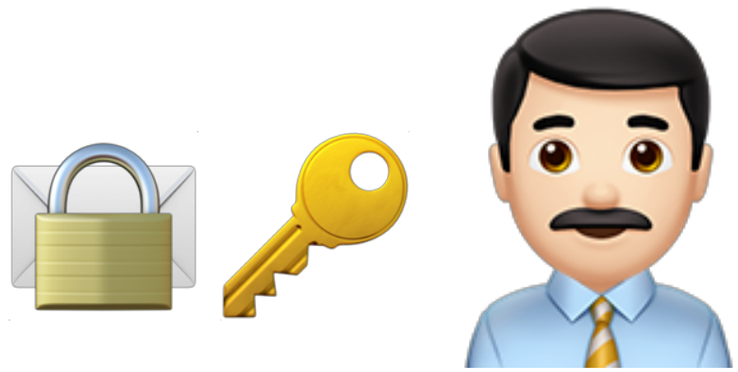
Bob

Eve

# Communications sécurisées



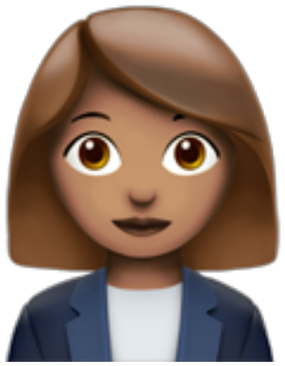
Alice



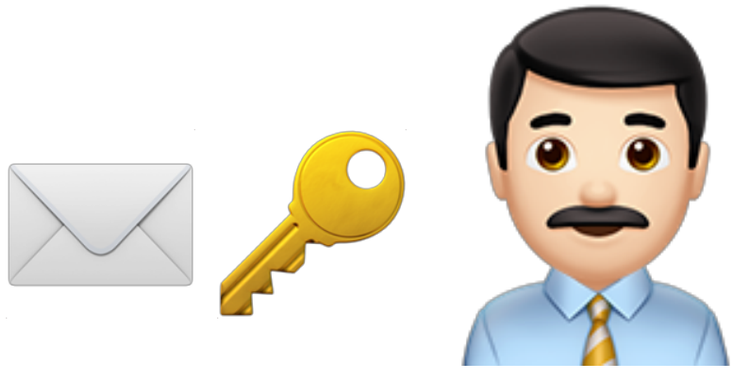
Bob

Eve

# Communications sécurisées



Alice



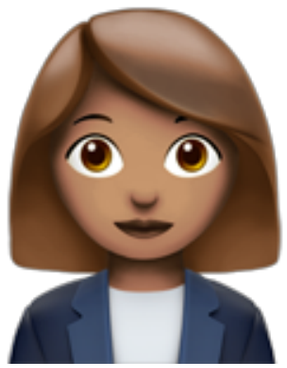
Bob

Eve

# On connait déjà

- Chiffrement de Cesar
- Chiffrement spartiate
- Chiffrement de Vigenère
- …mais ça ne suffit plus aujourd’hui

# Cryptosystème RSA

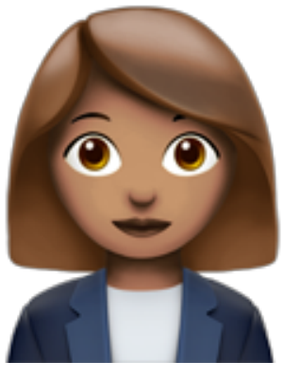


$(e, n)$ $(d, n)$

Alice Bob

$\in \{0, \ldots, n-1\}$

# Chiffrement et déchiffrement RSA



Alice

$(e, n)$

$$M \in \{0, \ldots, n-1\}$$

$$C = M^e \bmod n$$

Bob

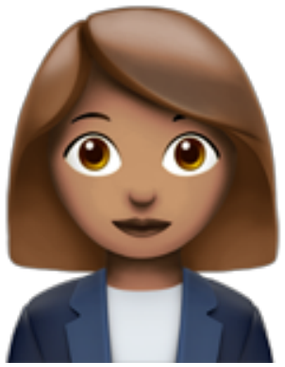
$(d, n)$

$$C^d \bmod n = M$$

# Comment choisir les clés RSA

1. Choisir $p$ et $q$ deux grands nombres premiers différents
2. Calculer $n = pq$
3. Calculer $\phi(n) = (p - 1)(q - 1)$
4. Choisir un entier $e$ premier avec $\phi(n)$
5. Calculer l'entier $d < \phi(n)$ tel que $de \bmod \phi(n) = 1$

# Théorème



Alice $(e, n)$

$$M \in \{0, \ldots, n-1\}$$

$$C = M^e \bmod n$$

Bob $(d, n)$

$$C^d \bmod n = M$$

# Calculer les puissances

$$x \quad x^2 \quad x^3 \quad \dots \quad x^n$$

# Calculer les puissances

$$x \quad x^2 \quad x^3 \quad \cdots \quad x^n$$

```
fonction puissance(x, n)
    y ≔ 1
    pour i ≔ 1 à n faire
        y ≔ yx
    retourner y
```

# Calculer les puissances

$$x \quad x^2 \quad x^3 \quad \cdots \quad x^n$$

```
fonction puissance(x, n)
    y ≔ 1
    pour i ≔ 1 à n faire
        y ≔ yx
    retourner y
```

***n*** **multiplications**

# Exponentiation rapide

$$x^{16}$$

# Exponentiation rapide

$$x^{16} = (x^8)^2$$

# Exponentiation rapide

$$x^{16} = (x^8)^2$$

$$= ((x^4)^2)^2$$

# Exponentiation rapide

$$x^{16} = (x^8)^2$$

$$= ((x^4)^2)^2$$

$$= (((x^2)^2)^2)^2$$

# Exponentiation rapide

$$x^{16} = (x^8)^2$$

$$= ((x^4)^2)^2$$

$$= (((x^2)^2)^2)^2$$

**4 multiplications**

# Exponentiation rapide

$$x^{13}$$

# Exponentiation rapide

$$x^{13} = (x^6)^2 \times x$$

# Exponentiation rapide

$$x^{13} = (x^6)^2 \times x$$
$$= ((x \times x \times x)^2)^2 \times x$$

# Exponentiation rapide

$$x^{13} = (x^6)^2 \times x$$

$$= ((x \times x \times x)^2)^2 \times x$$

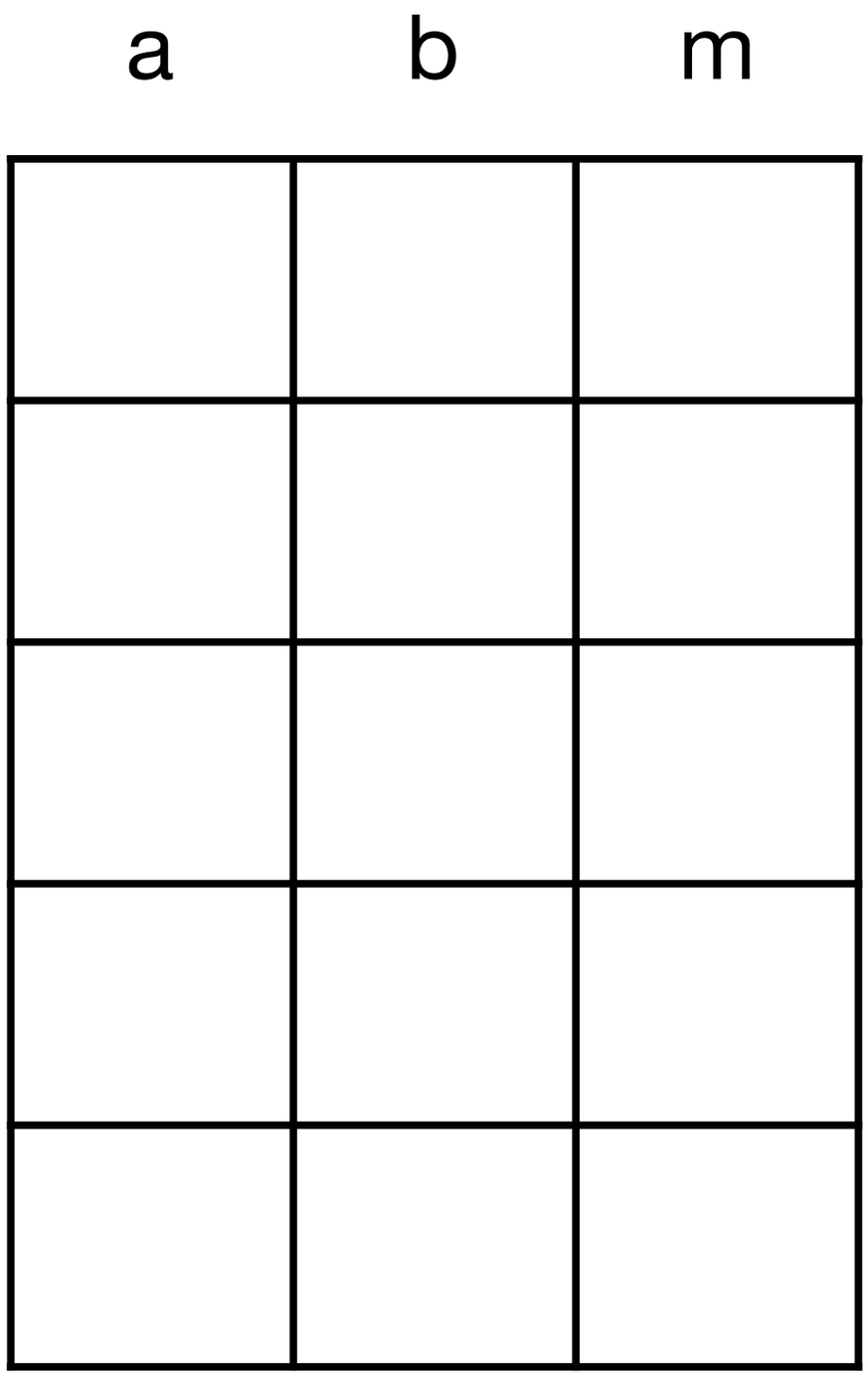
**5 multiplications**

# Exponentiation rapide

```
fonction puissance(x, n)
    a ≔ 1
    b ≔ x
    m ≔ n
    tant que m > 0 faire
        si m mod 2 = 0 alors
            m ≔ m / 2
        sinon
            m ≔ (m – 1) / 2
            a ≔ a × b
        b ≔ b × b
    retourner a
```

# Exponentiation rapide

```
fonction puissance(x, n)
    a ≔ 1
    b ≔ x
    m ≔ n
    tant que m > 0 faire
        si m mod 2 = 0 alors
            m ≔ m / 2
        sinon
            m ≔ (m – 1) / 2
            a ≔ a × b
        b ≔ b × b
    retourner a
```



# Exponentiation rapide

```
fonction puissance(x, n)
    a ≔ 1
    b ≔ x
    m ≔ n
    tant que m > 0 faire
        si m mod 2 = 0 alors
            m ≔ m / 2
        sinon
            m ≔ (m – 1) / 2
            a ≔ a × b
        b ≔ b × b
    retourner a
```

| a | b | m |
|---|---|---|
| 1 | x | 13 |
| | | |
| | | |
| | | |
| | | |

# Exponentiation rapide

```
fonction puissance(x, n)
    a ≔ 1
    b ≔ x
    m ≔ n
    tant que m > 0 faire
        si m mod 2 = 0 alors
            m ≔ m / 2
        sinon
            m ≔ (m – 1) / 2
            a ≔ a x b
        b ≔ b x b
    retourner a
```

| a | b | m |
|---|---|---|
| 1 | x | 13 |
| x | $x^2$ | 6 |
| | | |
| | | |
| | | |

# Exponentiation rapide

```
fonction puissance(x, n)
    a ≔ 1
    b ≔ x
    m ≔ n
    tant que m > 0 faire
        si m mod 2 = 0 alors
            m ≔ m / 2
        sinon
            m ≔ (m – 1) / 2
            a ≔ a x b
        b ≔ b x b
    retourner a
```

| a | b | m |
|---|---|---|
| 1 | x | 13 |
| x | $x^2$ | 6 |
| x | $x^4$ | 3 |
| | | |
| | | |

# Exponentiation rapide

```
fonction puissance(x, n)
    a ≔ 1
    b ≔ x
    m ≔ n
    tant que m > 0 faire
        si m mod 2 = 0 alors
            m ≔ m / 2
        sinon
            m ≔ (m – 1) / 2
            a ≔ a × b
        b ≔ b × b
    retourner a
```

| a | b | m |
|---|---|---|
| 1 | x | 13 |
| x | $x^2$ | 6 |
| x | $x^4$ | 3 |
| $x^5$ | $x^8$ | 1 |
| | | |

# Exponentiation rapide

```
fonction puissance(x, n)
    a ≔ 1
    b ≔ x
    m ≔ n
    tant que m > 0 faire
        si m mod 2 = 0 alors
            m ≔ m / 2
        sinon
            m ≔ (m – 1) / 2
            a ≔ a × b
        b ≔ b × b
    retourner a
```

| a | b | m |
|---|---|---|
| 1 | x | 13 |
| x | $x^2$ | 6 |
| x | $x^4$ | 3 |
| $x^5$ | $x^8$ | 1 |
| $x^{13}$ | $x^{16}$ | 0 |

# Exponentiation rapide

```
fonction puissance(x, n)
    a ≔ 1
    b ≔ x
    m ≔ n
    tant que m > 0 faire
        si m mod 2 = 0 alors
            m ≔ m / 2
        sinon
            m ≔ (m – 1) / 2
            a ≔ a × b
        b ≔ b × b
    retourner a
```

| a | b | m |
|---|---|---|
| 1 | x | 13 |
| x | $x^2$ | 6 |
| x | $x^4$ | 3 |
| $x^5$ | $x^8$ | 1 |
| $x^{13}$ | $x^{16}$ | 0 |

# Calcul de fonctions mathématiques

# Racine carré

$$x = \sqrt{a}$$

$$x^2 = a \quad (x \geq 0)$$

$$x^2 - a = 0 \quad (x \geq 0)$$

# Racine carré

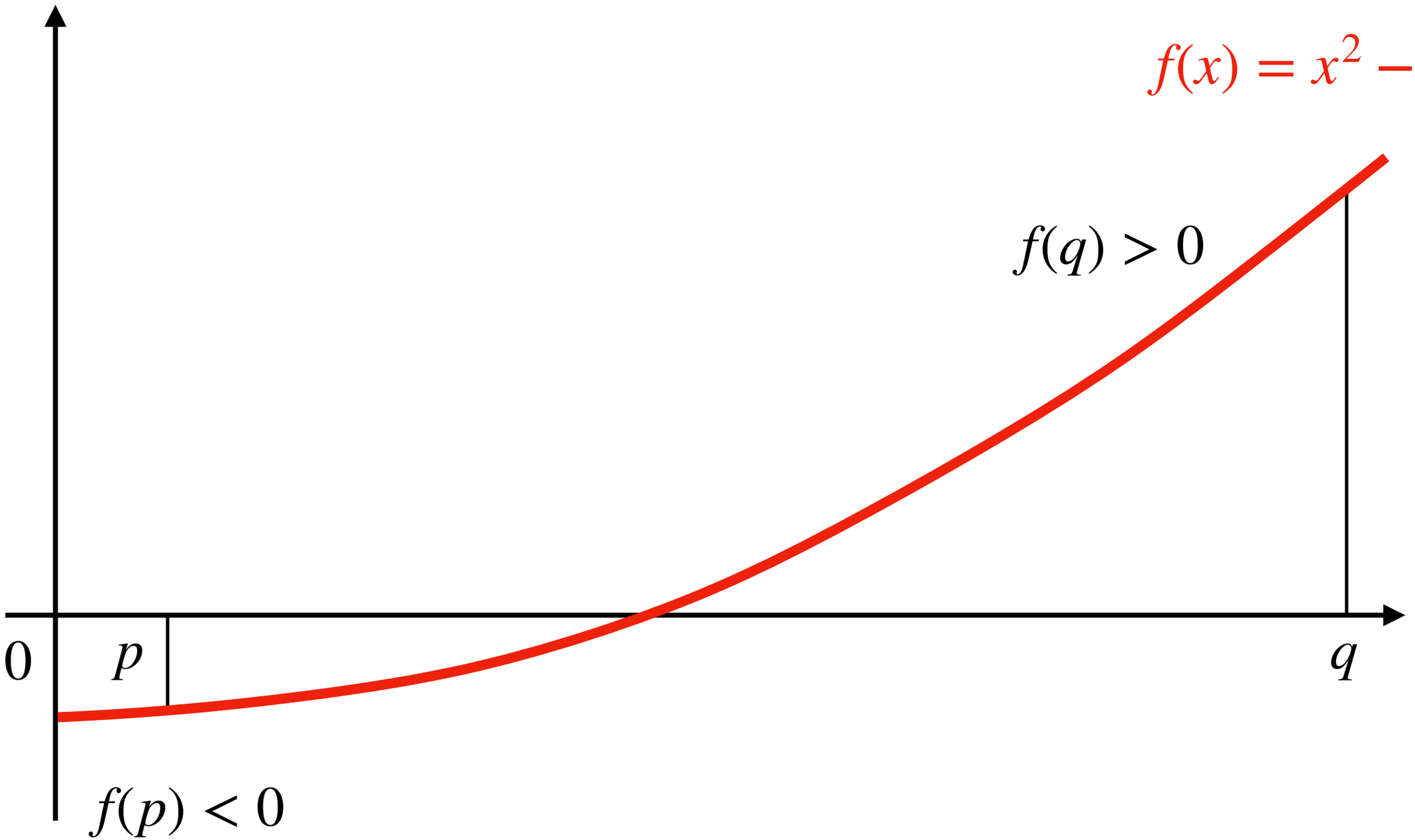
$$x = \sqrt{a}$$

$$\Updownarrow$$

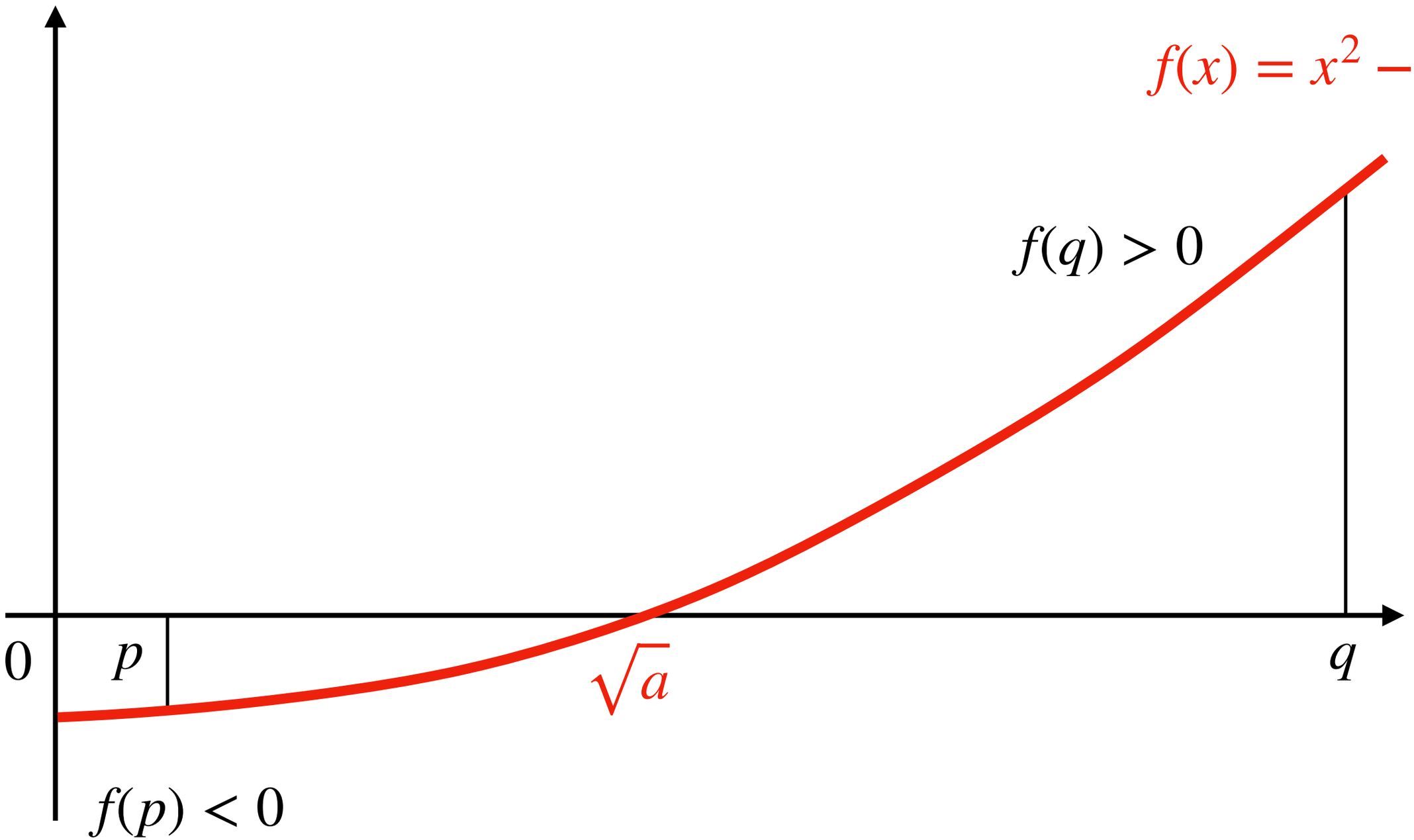
$$f(x) = 0 \qquad \text{où} \qquad f(x) : \mathbb{R}_{\geq 0} \to \mathbb{R}$$

$$f(x) = x^2 - a$$

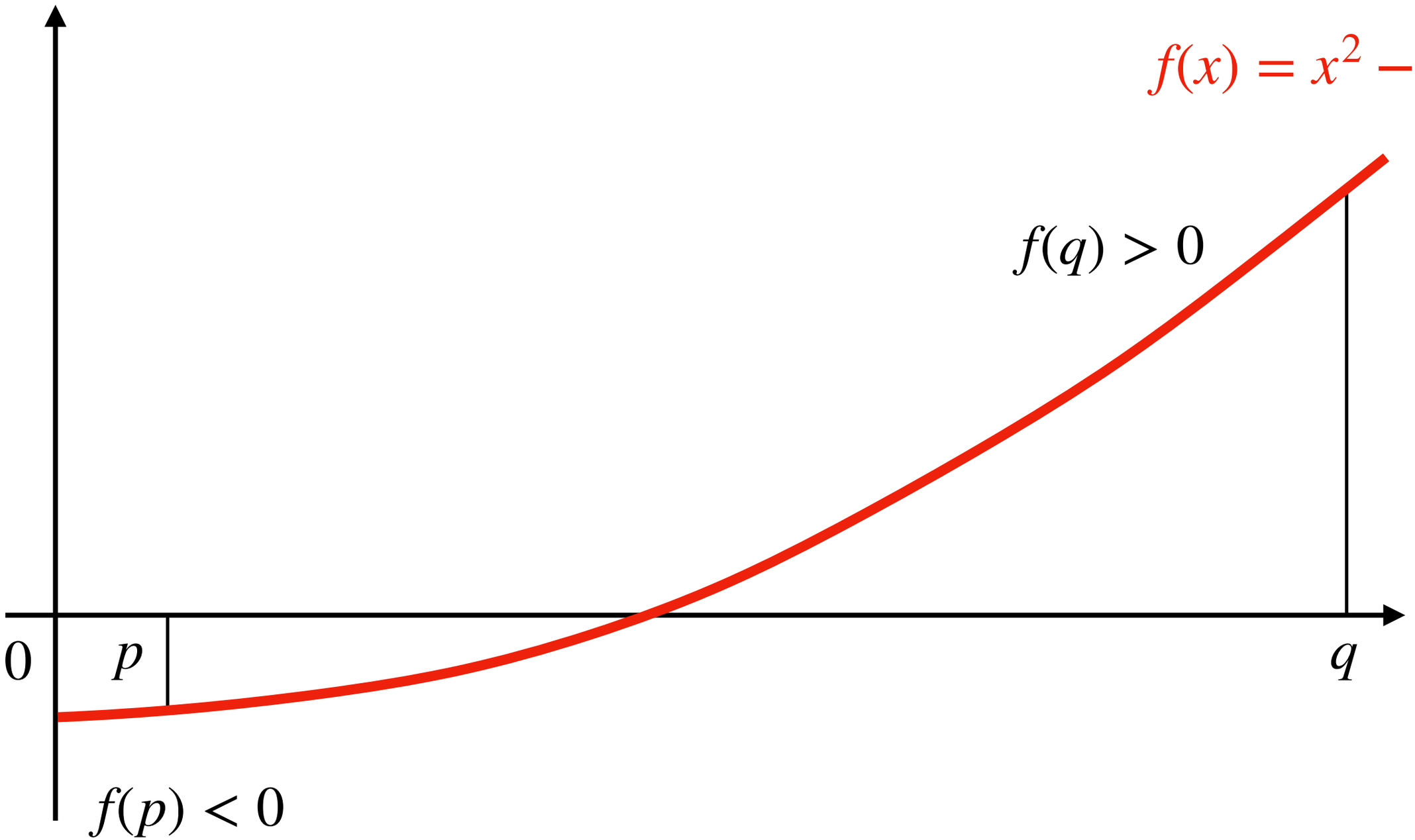
# Approximations des zéros par recherche dichotomique



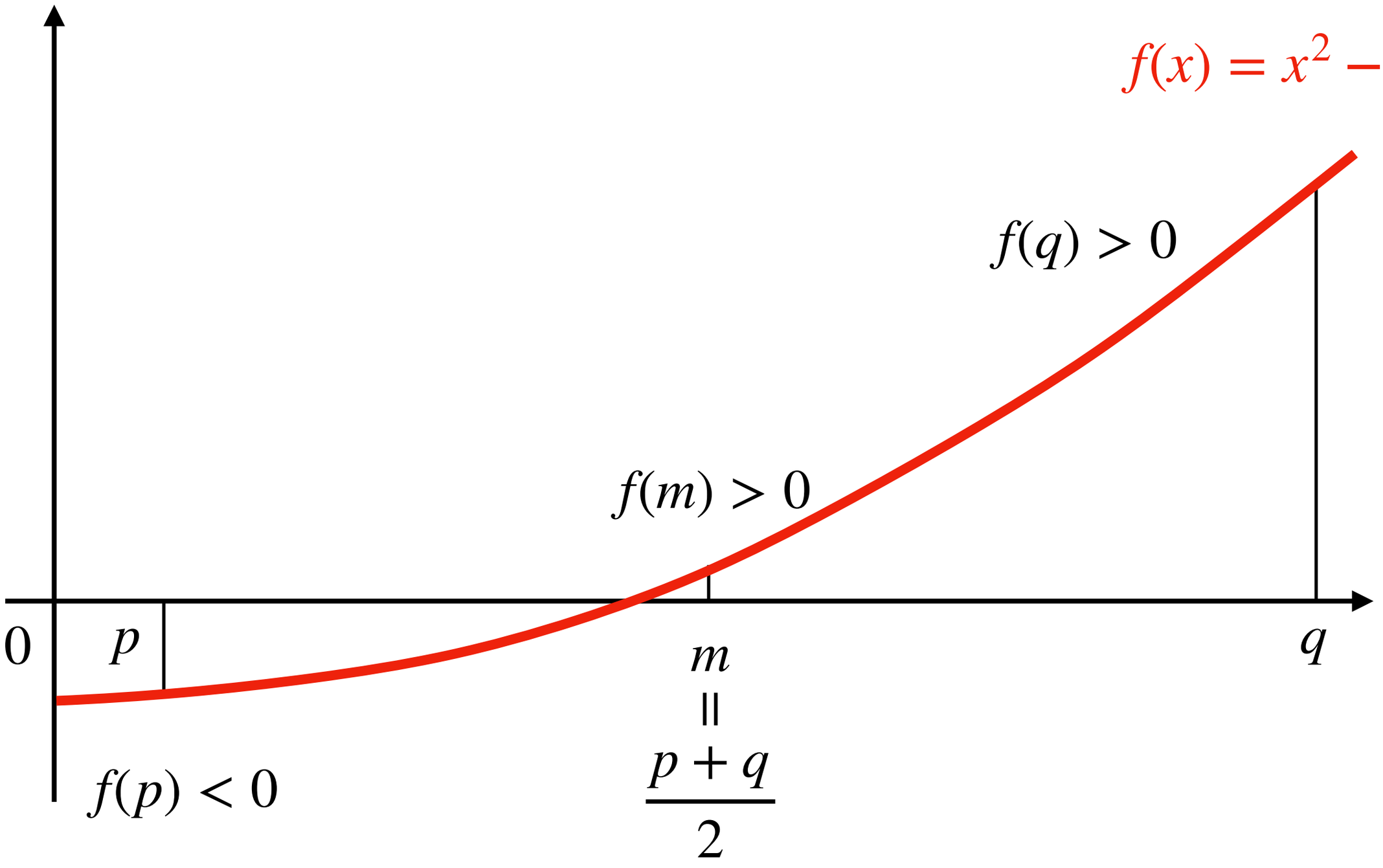
# Approximations des zéros par recherche dichotomique



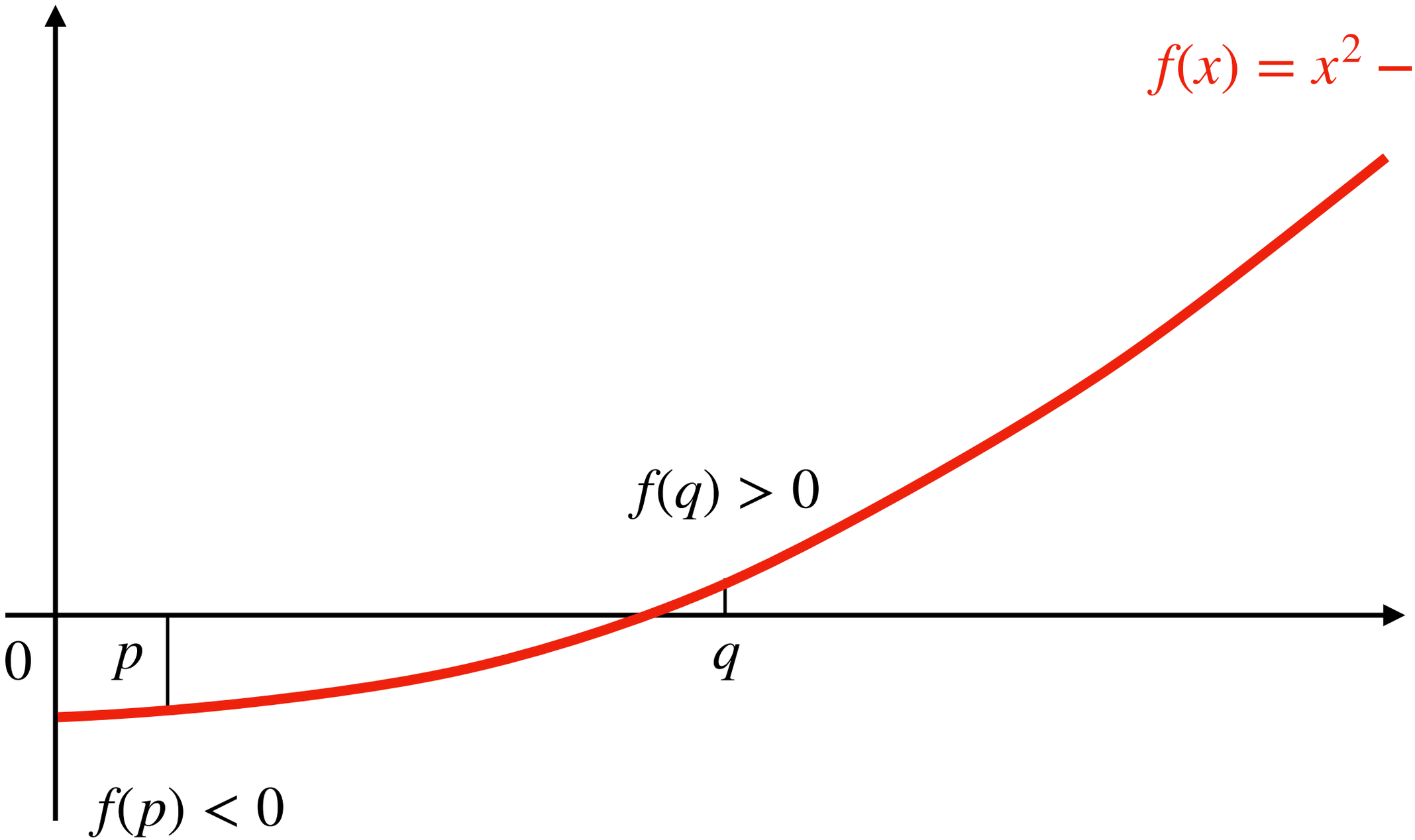
# Approximations des zéros par recherche dichotomique



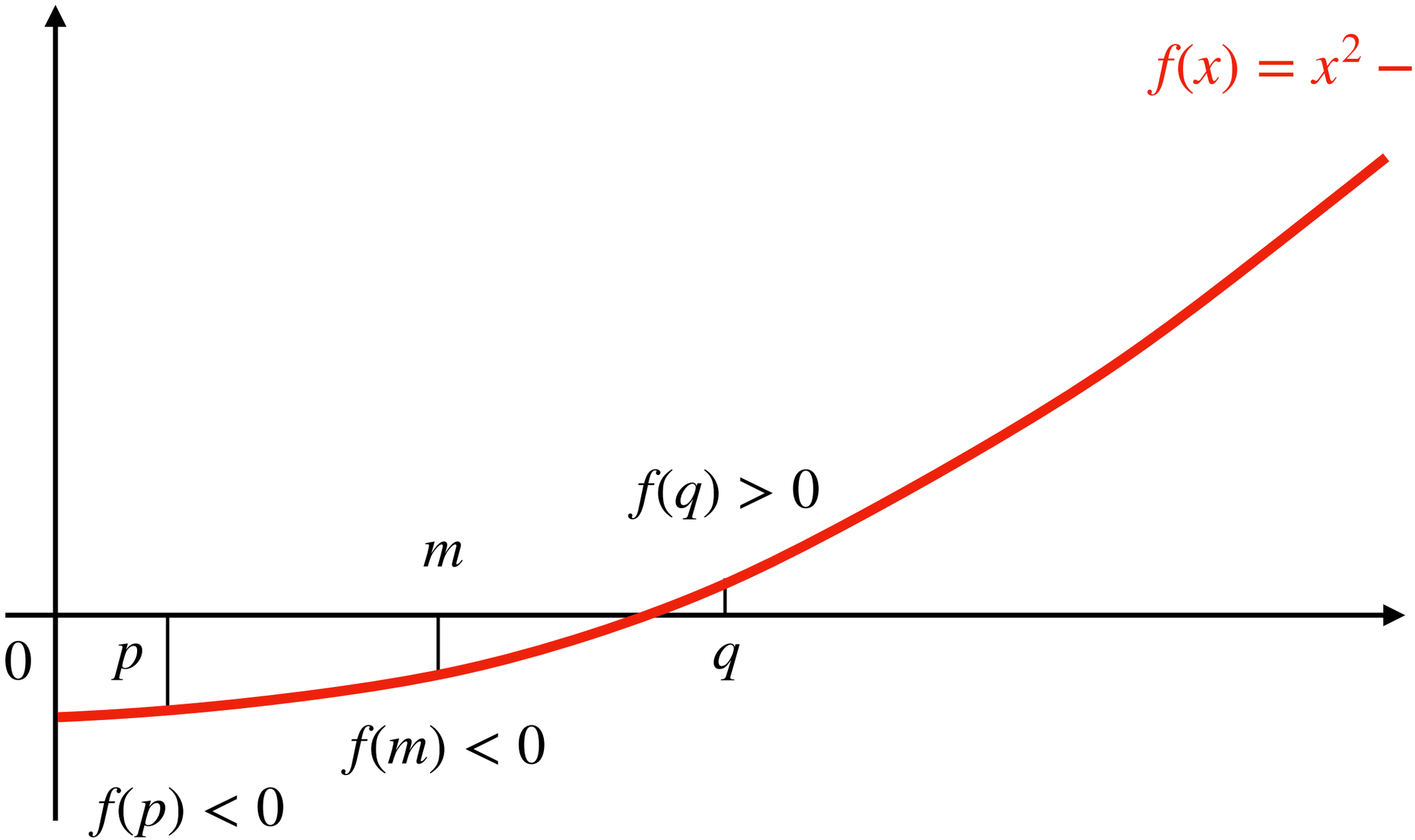
# Approximations des zéros par recherche dichotomique



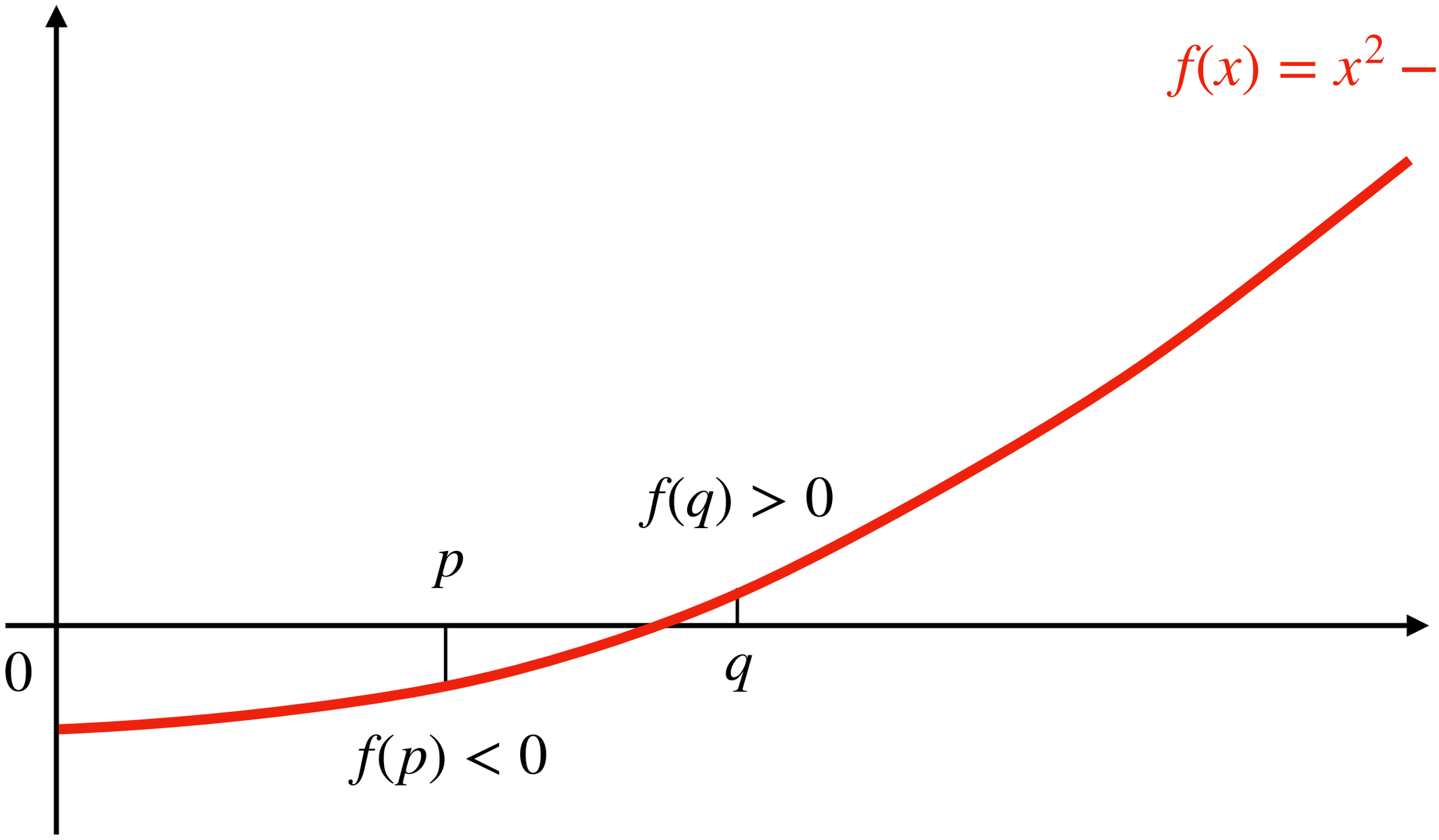
# Approximations des zéros par recherche dichotomique



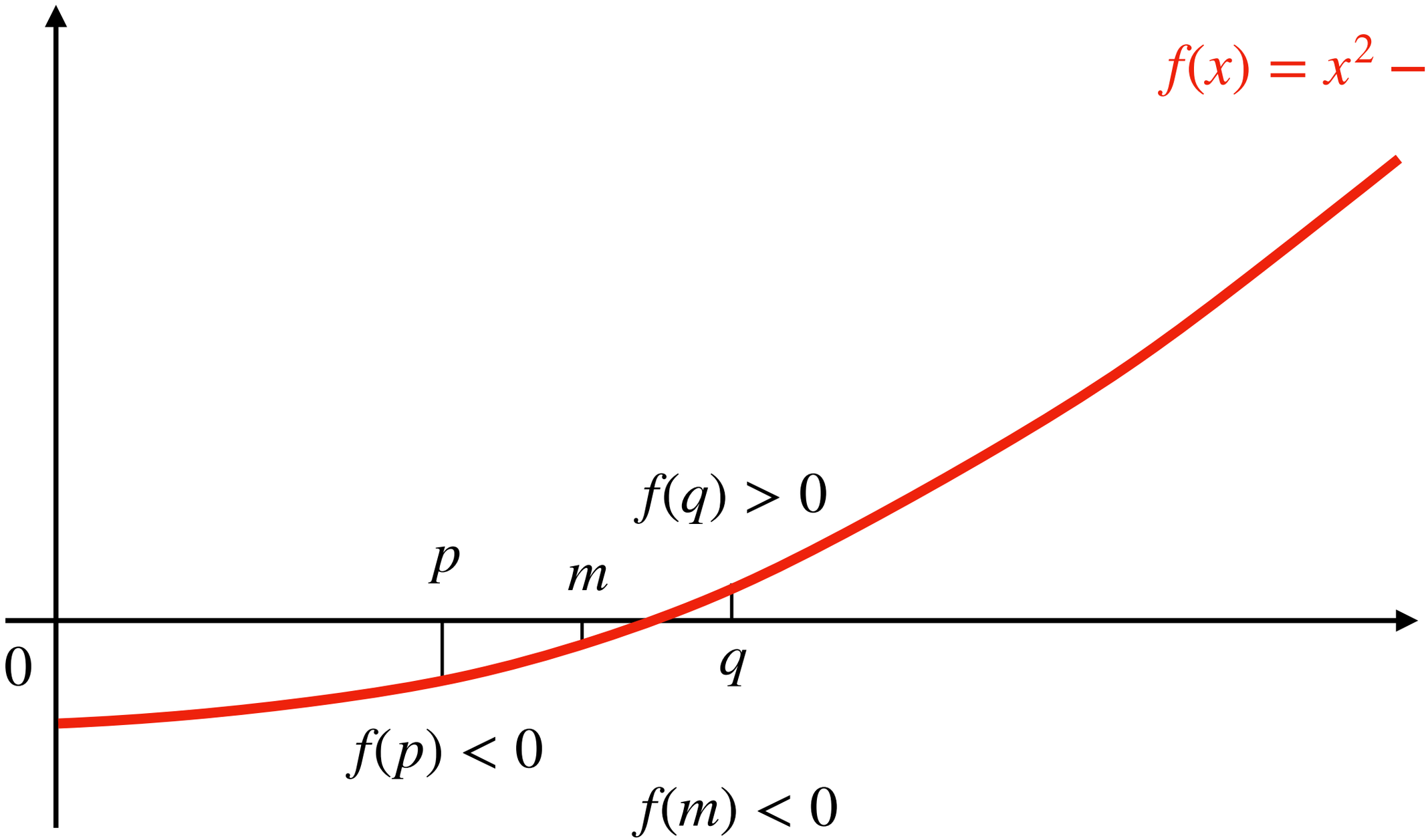
# Approximations des zéros par recherche dichotomique



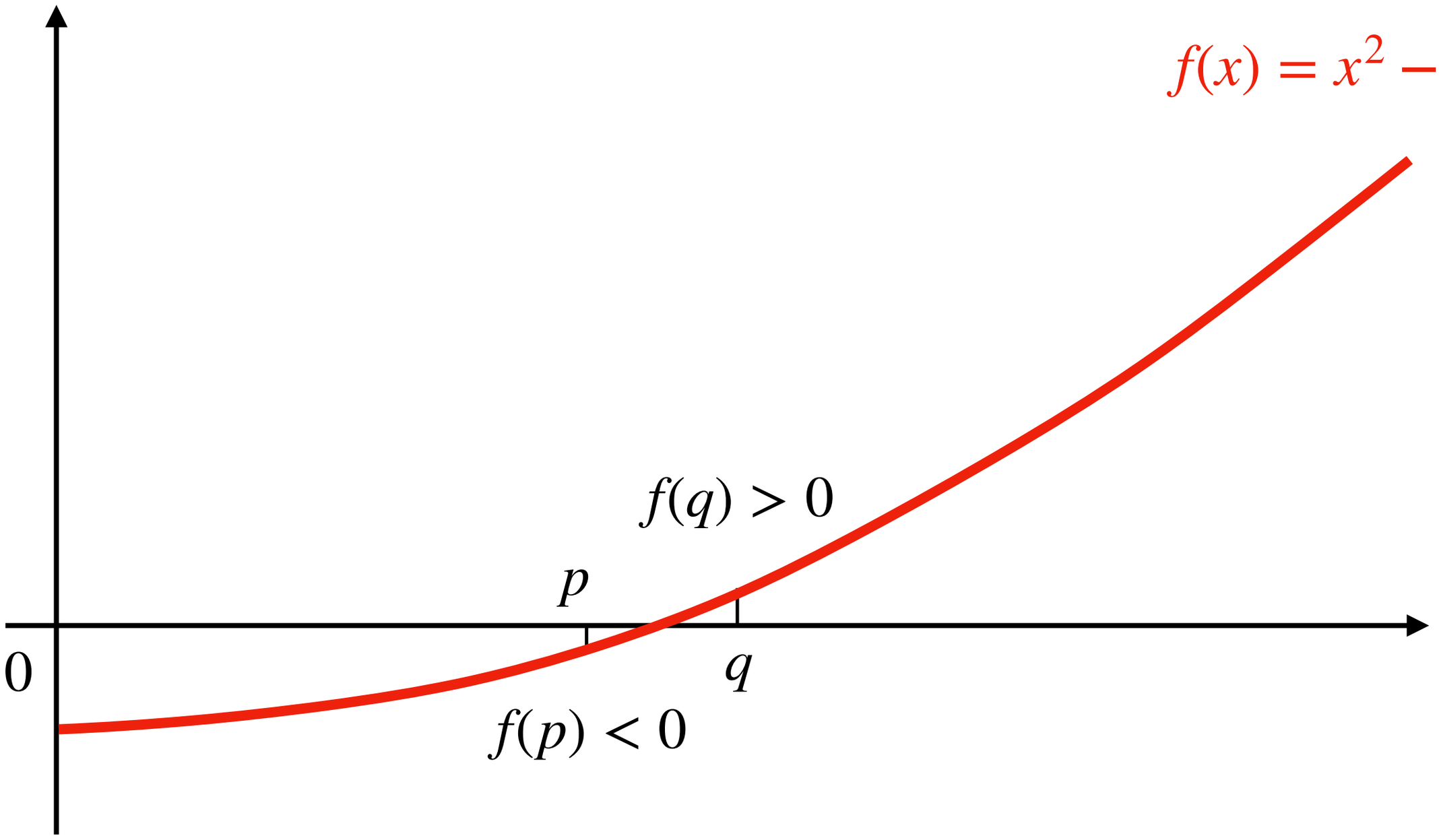
# Approximations des zéros par recherche dichotomique



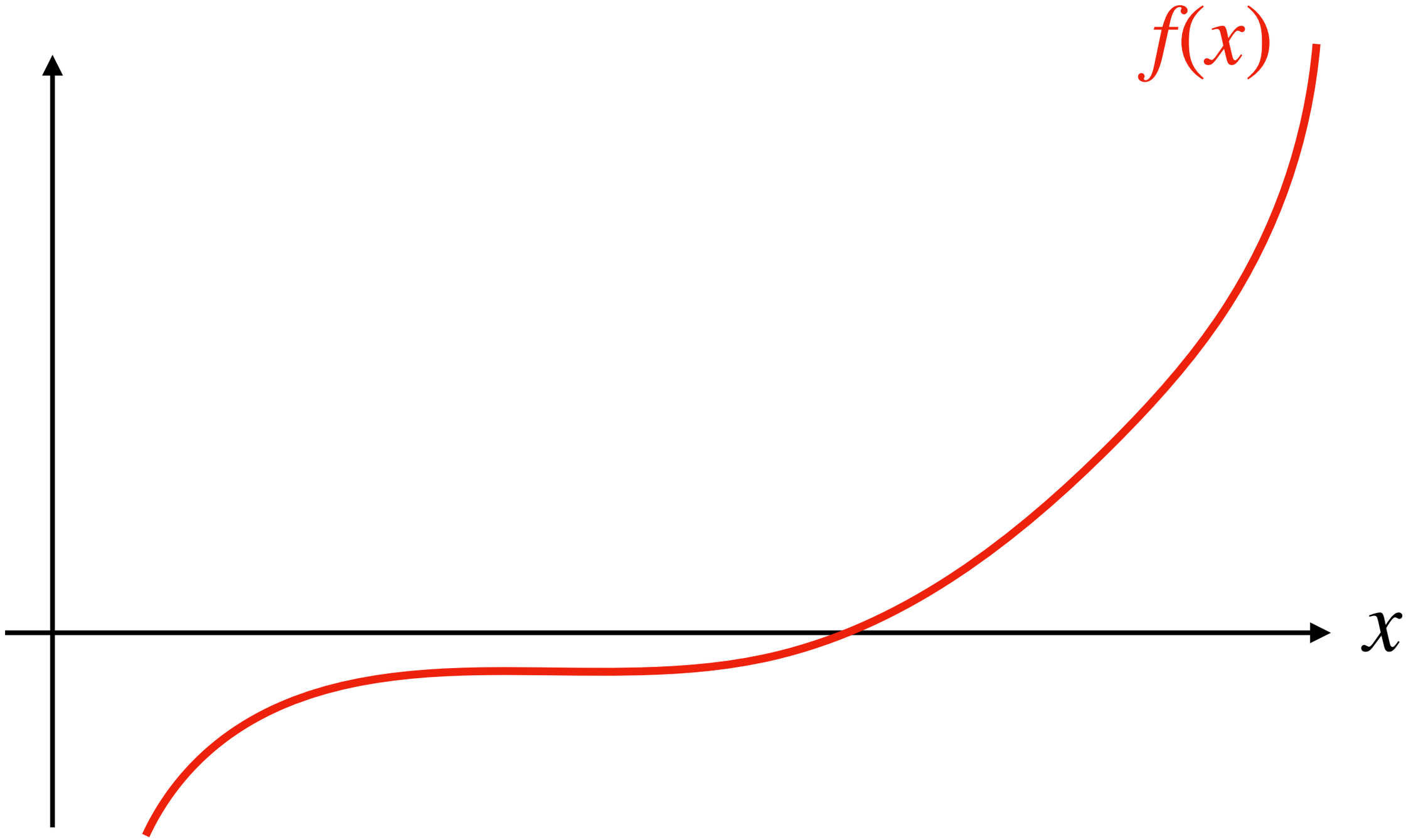
# Approximations des zéros par recherche dichotomique



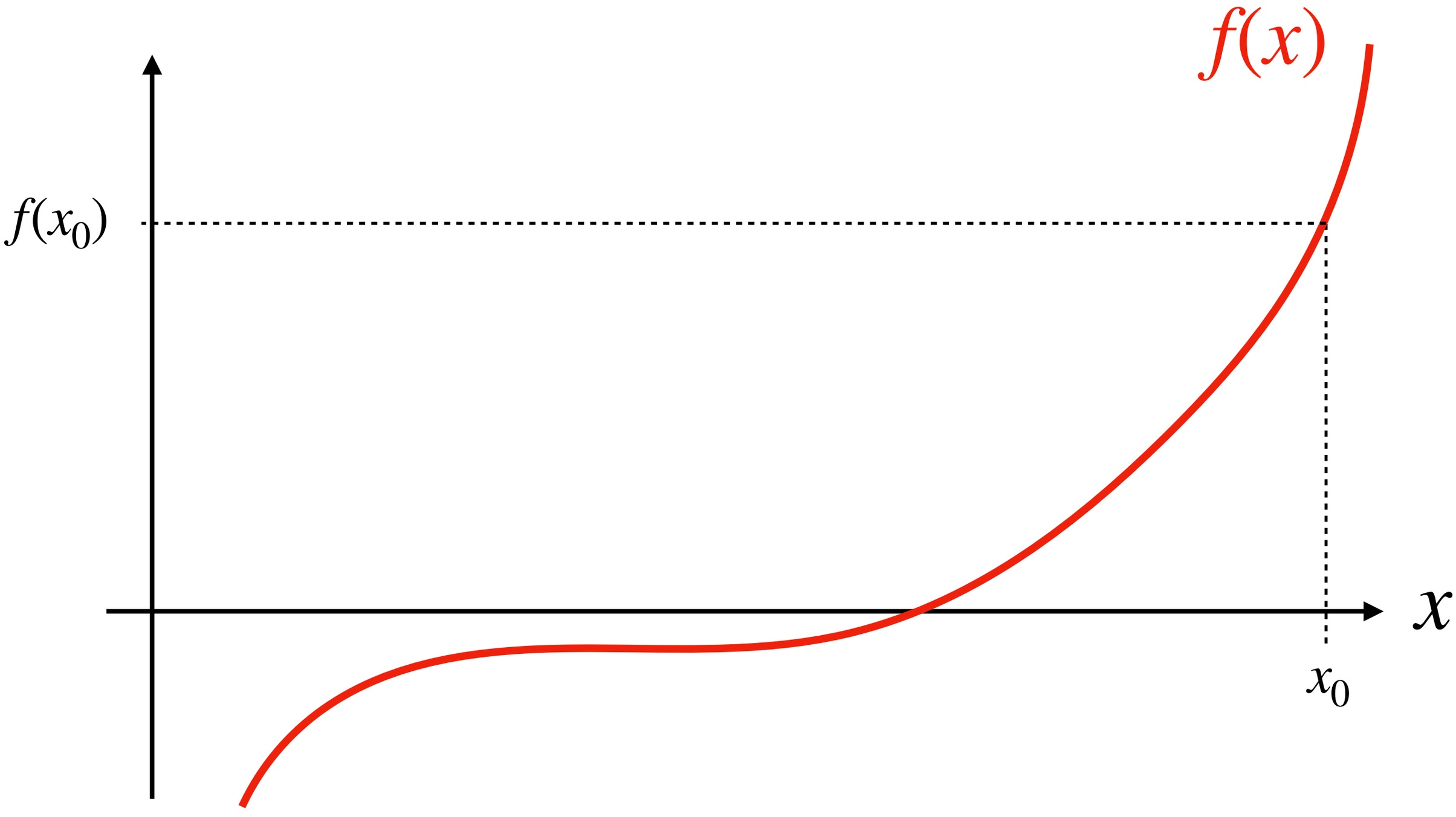
# Approximations des zéros par recherche dichotomique



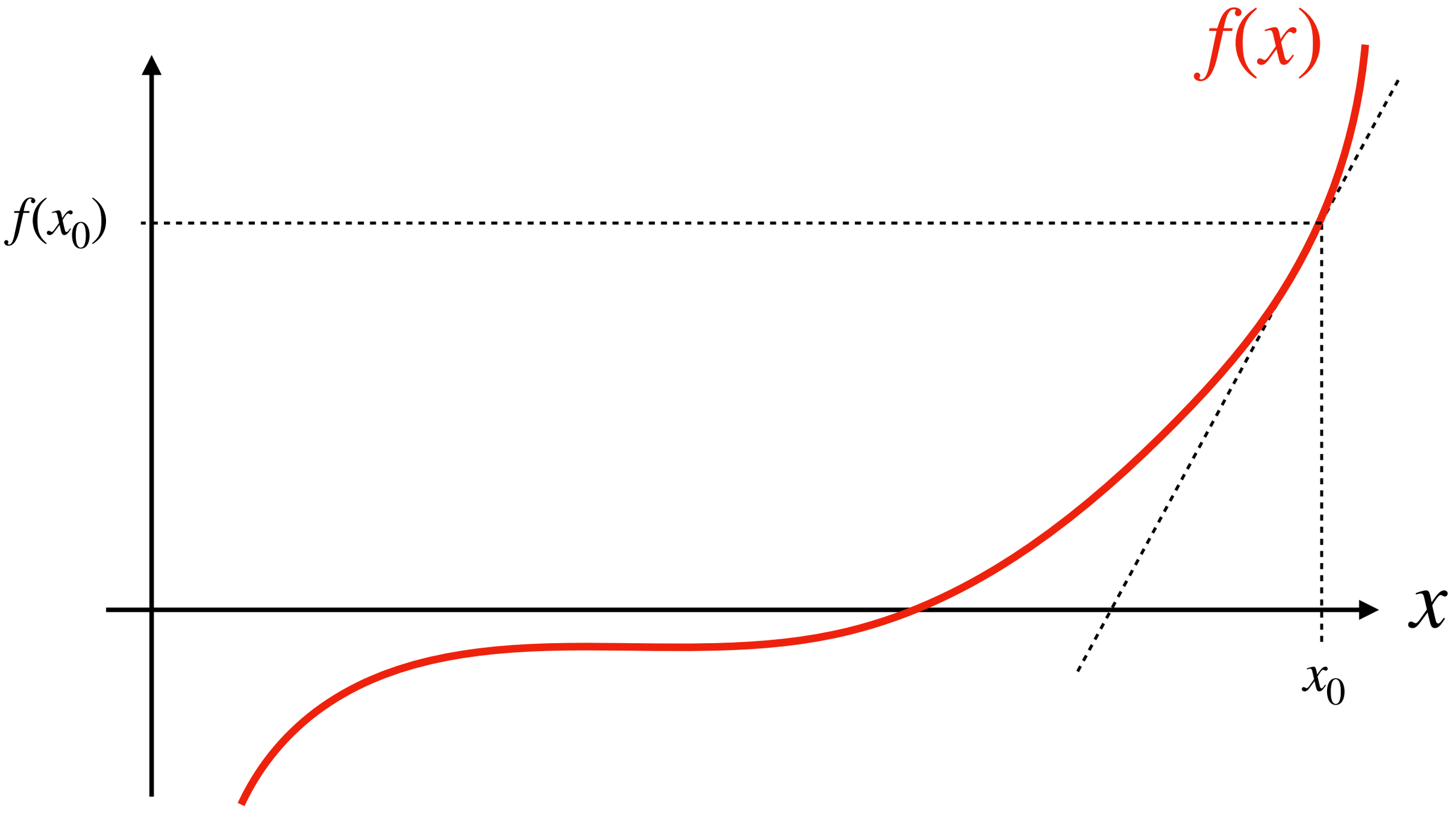
# Méthode de la descente



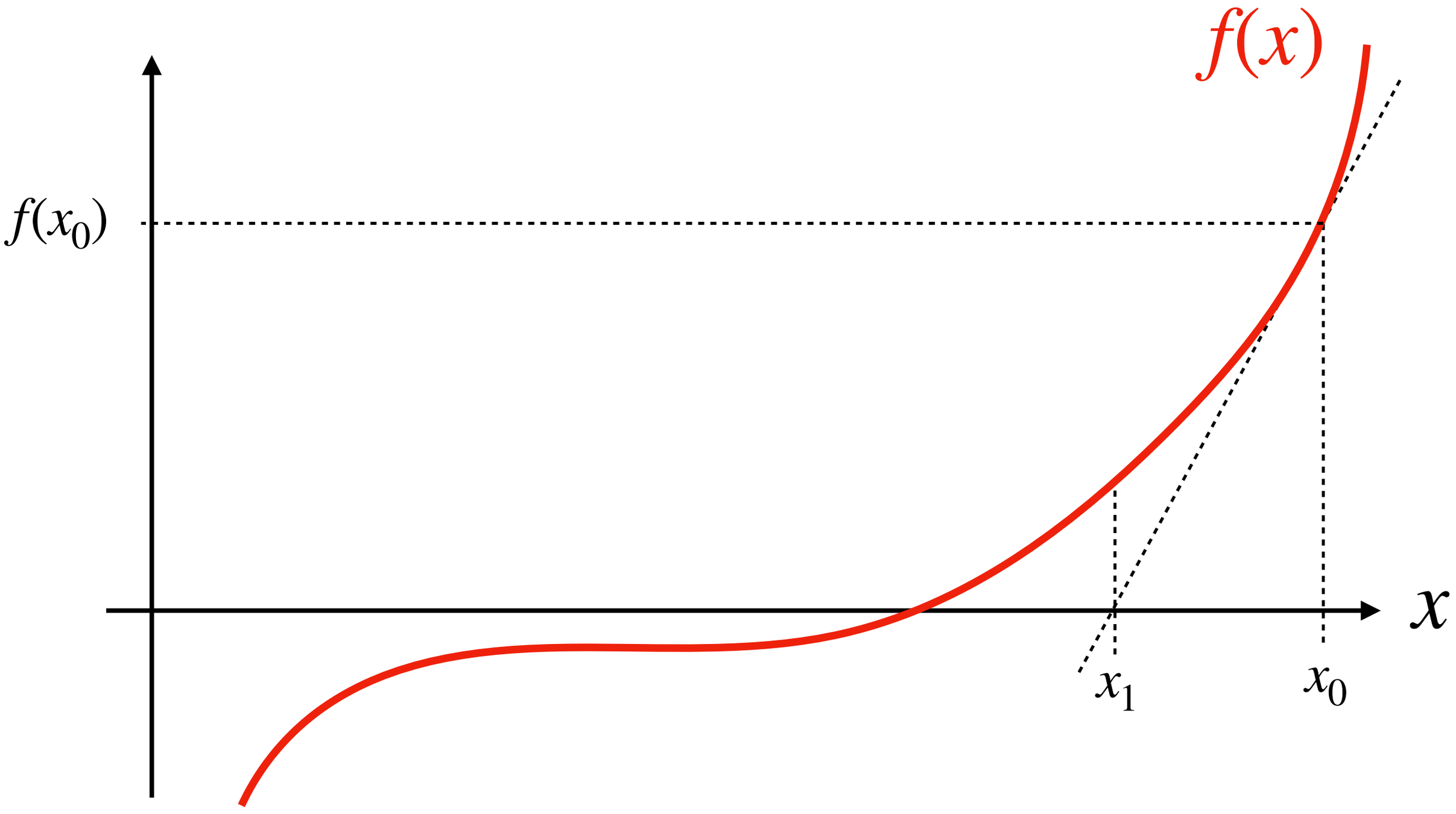
# Méthode de la descente



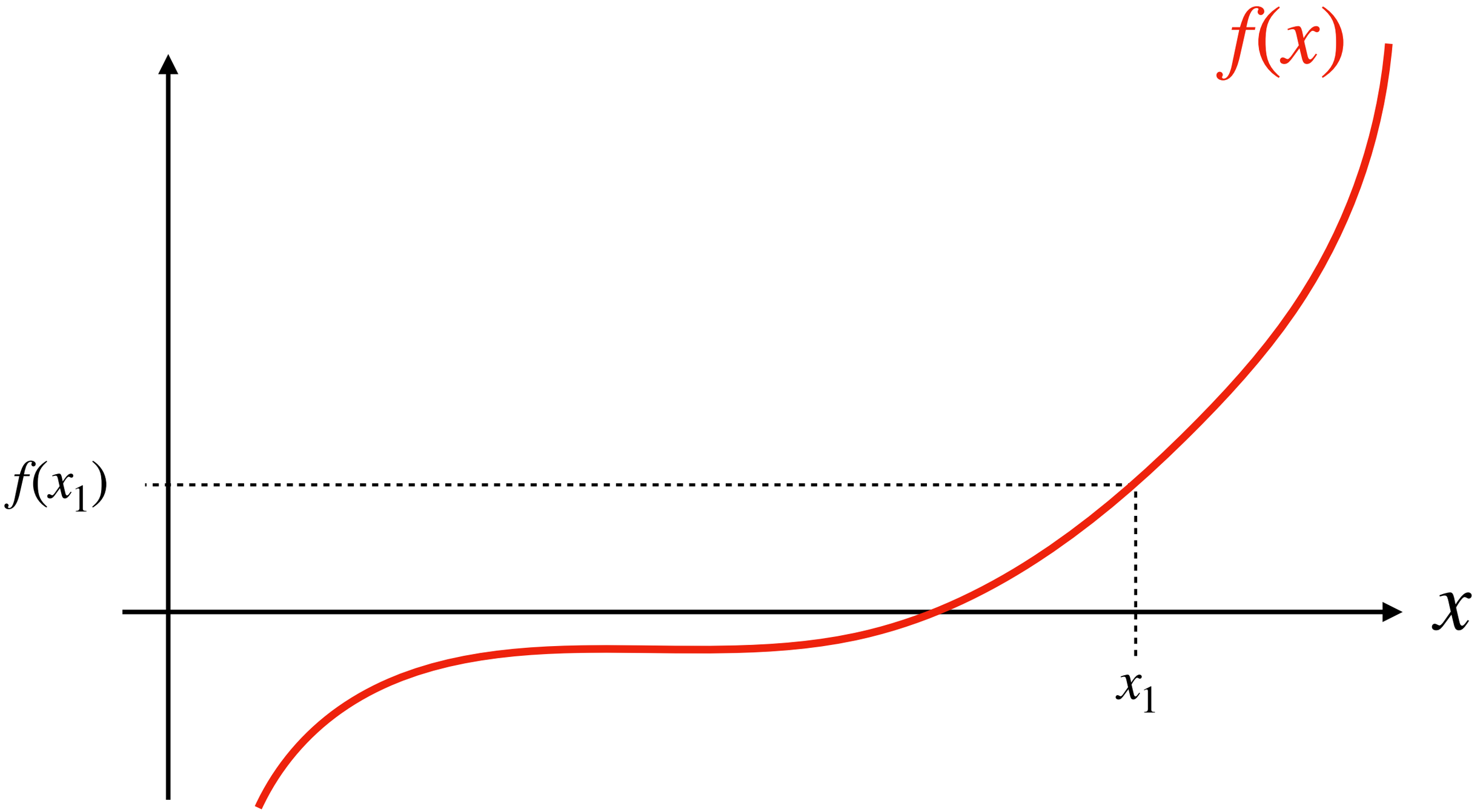
# Méthode de la descente



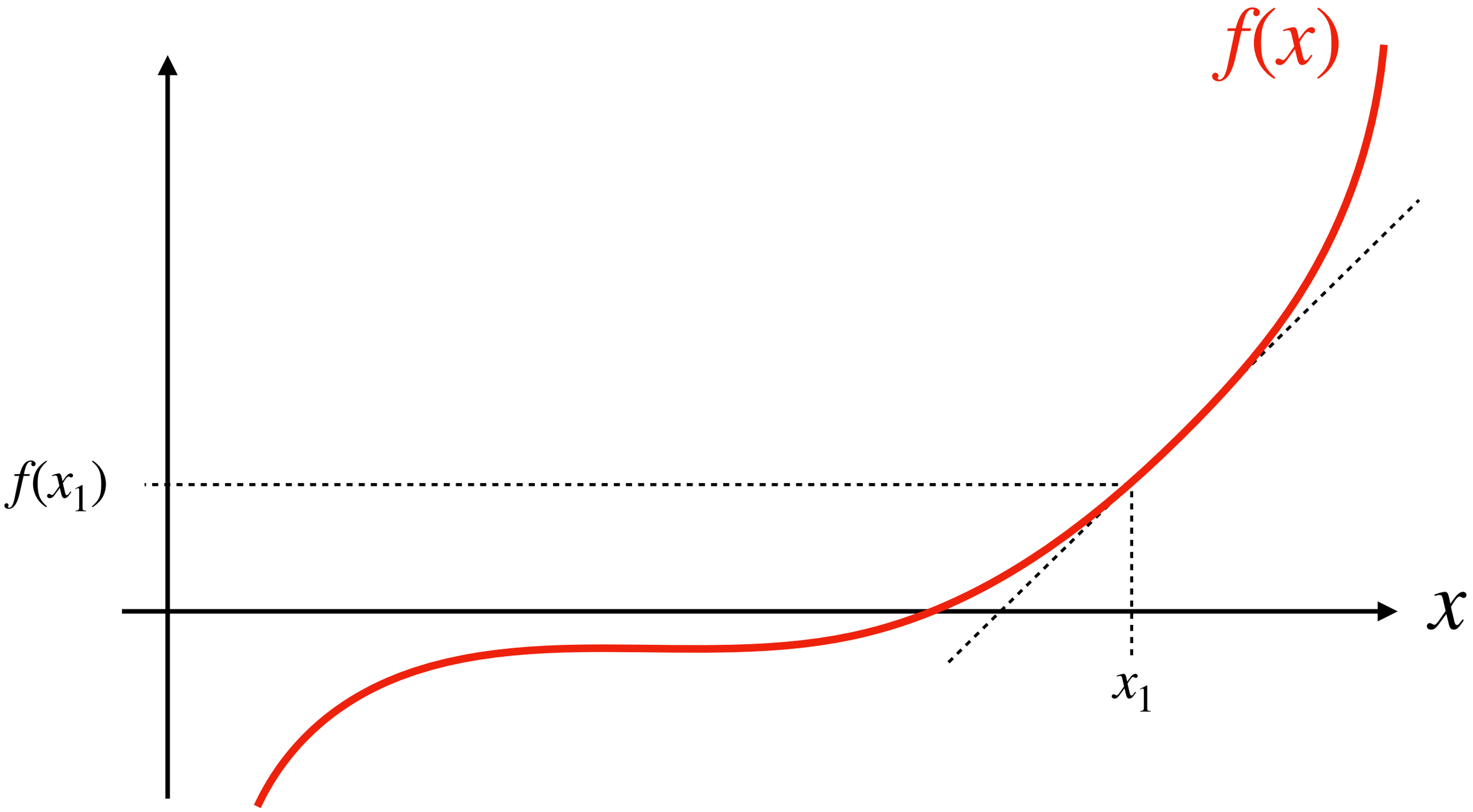
# Méthode de la descente



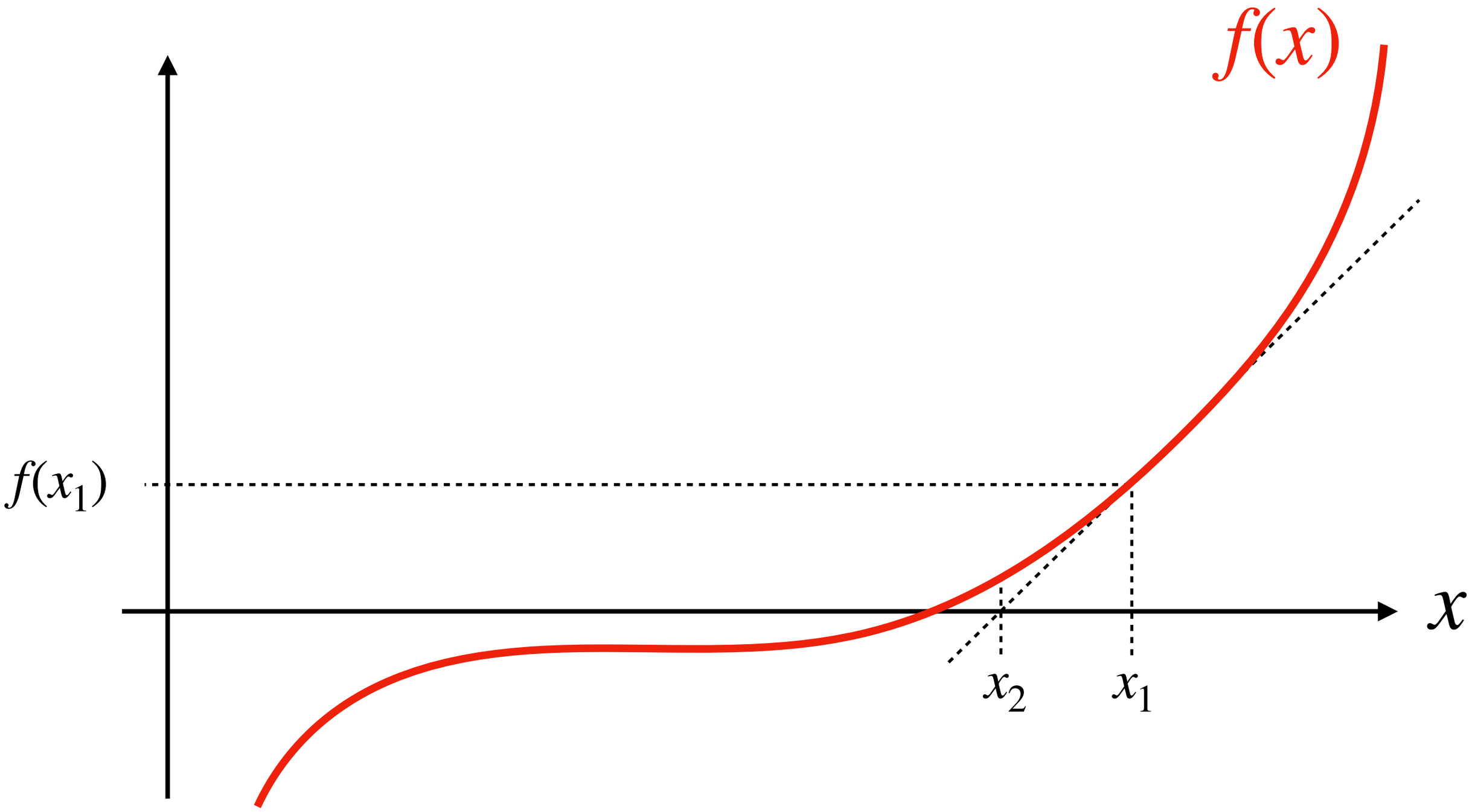
# Méthode de la descente



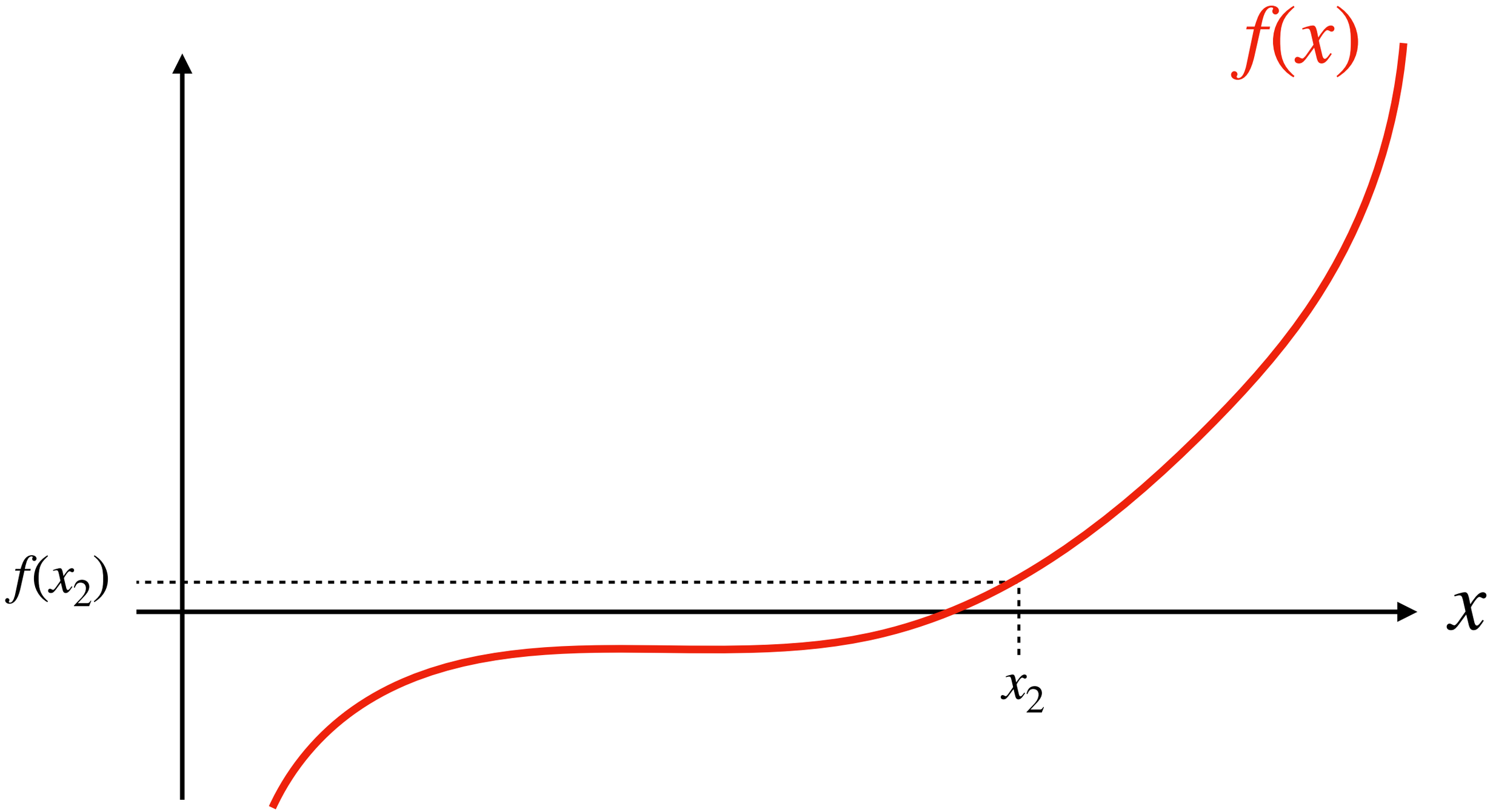
# Méthode de la descente



# Méthode de la descente



# Méthode de la descente



# Algorithme de Newton

**fonction** approx-zéro(f, $x_0$)
    a ≔ $x_0$
**tant que** f(a) ≠ 0 **faire**
        tracer la tangente t en a
        a ≔ abscisse de l'intersection de t
                et de l'axe des abscisses
**retourner** a

# Tracer la tangente

- La tangente à *f* en *a* est la droite d'équation

$$y = f(a) + (x - a)\, f'(a)$$

- On a $y = 0$ quand

$$x = a - \frac{f(a)}{f'(a)}$$

# Algorithme de Newton

**fonction** approx-zéro(f, $x_0$)
a ≔ $x_0$
**tant que** f(a) ≠ 0 **faire**
    a ≔ a – f(a) / f’(a)
**retourner** a

# Algorithme de Newton

**fonction** approx-zéro(f, f’, $x_0$)
a ≔ $x_0$
**tant que** f(a) ≠ 0 **faire**
    a ≔ a – f(a) / f’(a)
**retourner** a

# Terminaison

**fonction** approx-zéro(f, f’, $x_0$, ε)
a ≔ $x_0$
**tant que** |f(a)| < ε **faire**
    a ≔ a – f(a) / f’(a)
**retourner** a